Appendix C: COE Tools

This appendix lists the COE tools available to assist developers in creating, installing, and testing segments. This appendix is not intended to provide an exhaustive discussion of all tool features. Additional features and improvements in the tools may be made before updates to this document are released.

The previous *I&RTS* provided more detailed information on parameters the tools accept, and their syntax. This level of detail has been removed from this version of the *I&RTS* and moved to Developer's Toolkit documentation. Backwards compatibility has been preserved, but there are new tools with this release, and some of the tools accept additional parameters to provide new functionality. Refer to the Developer's Toolkit release notes, the online man pages for the tools, and the help page provided by the tools for the latest information.

C-1. General Tool Features

Most of the tools are designed to run interactively, and accept one or more command-line parameters. Many of the tools are designed to be used in the installation process during `PostInstall` or `PreInstall`, while others are intended to be used during the development process. The development tools may be invoked individually from a command line, or from an Integrated Development Environment tool, `COEIDE`.

Tools that begin with the prefix `COE` are runtime tools[1] and are thus distributed to operational sites. Those which do not begin with the prefix `COE` are developer tools only and are distributed as part of the Developer's Toolkit, but are not normally sent to operational sites. This section describes other conventions that are common to the tools.

C-1.1 Communicating with the Tools

The COE tools can be accessed in two different ways. First, a user program can be written and linked with the tools library.[2] Such a program goes through the defined APIs to request services from the tools. A second way that the tools can be used is to execute the tools as a separate program and use services provided by the operating system to communicate between the tool and the calling program. This technique is most frequently used in as a way to invoke the tools from within a `PostInstall` script. Whichever approach is most convenient may be used. All of the tools are available as individual executables, but not all are available through a library interface.

When accessed through a library interface, the tools return data through the interface. They may also write to `stdout`. When used as separate executable programs, the tools communicate with the outside world in two ways.

1. The tools use the `exit` function to set the Unix `status` environment variable. (`status` is used for the C shell `csh`. The `$?` environment variable is used for POSIX shells.) `status` is set to 0 for normal tool completion. A `status` return value other than 0 indicates a completion code that is tool-specific. Refer to the appropriate Developer's Toolkit documentation on man pages for a complete description of the tools and their return codes, including error return codes.[3]

2. The tools use `stdin`, `stdout`, and `stderr` and thus support I/O redirection. Redirecting `stdin` allows the tools to receive input from a file or another program, while redirecting `stdout` allows the tools to provide input to other programs. (Redirecting `stdin` is not always convenient. The `-R` command-line parameter, see

[1] The one exception to the convention of using the prefix `COE` for runtime tools is `COEIDE`. `COEIDE` is intended only for the development process and would not normally be distributed to operational sites.
[2] Libraries for NT platforms are implemented as DLLs (Dynamic Link Libraries).
[3] The previous *I&RTS* indicated that the tools return a value of -1 to indicate an error. This convention has been preserved to the extent possible, but note that certain operating systems return a single byte as the `status` environment variable and hence return a value of 255 instead of -1. Refer to the appropriate Developer's Toolkit documentation or online man pages for complete documentation of tool return codes.

below, allows a tool to read input from a response file instead of `stdin`.) For example, the tool `COEPrompt` displays a message to the user and allows the user to type in a response, and the user's response is written to `stdout`. The tools write normal output to `stdout`, while errors and warnings are sent to `stderr`.

For Unix, the following statement illustrates how the second technique can be used to ask the user to enter the name of a file:

```
COEPrompt "Enter Filename" | MyProg
```

Or, the following can be used to write the results to a file:

```
COEPrompt "Enter Filename" >& /tmp/tempfile
```

Note that the above example redirects both `stdout` and `stderr` to `/tmp/tempfile`.

In the NT environment,

```
COEPrompt "Enter Filename" 1> C:\temp\tempfile 2>1&
```

will redirect output to the file `C:\temp\tempfile`.

C-1.2 Standard Parameters

The command-line parameters listed below are common to all tools for which they are meaningful:

`-C file`	Read command-line parameters from *file*.
`-h, H`	Display online help describing how to use the tool.
`-p path`	Use *path* to establish the path for subsequent file names.
`-R file`	Use *file* to respond to questions from the tool.
`-v`	Toggle the "verbose" flag. When enabled, print out diagnostic information as the tool executes. The initial state of the flag is disabled.
`-V`	Display the tool's version number.
`-w`	Toggle the "warnings" flag. When enabled, print out warning messages as the tool executes. The initial state of this flag is enabled.

The parameter `-p` can appear multiple times on the command line. For example,

```
tool -p /h/tst -C cmds -p ../tmp -R responses -p /h/tst2 -C cmds
```

uses the command-line parameter `-p` to specify the location of three files: `/h/tst/cmds`, `/h/tmp/responses`, and `/h/tst2/cmds`. The default path for files when the parameter `-p` does not appear is tool-specific, but is usually `/h`.

C-1.3 Tool Helper Functions

The DII COE tools are designed to be extensible to accommodate new capabilities as new technologies are employed, such as CORBA or DCE. To provide this extensibility, the tools are designed to accept *helper functions*. Helper functions are "bolt ons" invoked by the tools whenever they encounter a segment descriptor that they are unable to process. For example, the database-related segment descriptors are processed by helper functions. When a tool, such as `VerifySeg`, encounters a database-related descriptor, it invokes the helper function to process (e.g., validate) the descriptor. The DCE, Web, and Database-related descriptors are processed by helper functions.

Helper functions require prior authorization from the DII COE Chief Engineer before they can be used with any of the tools in this appendix. Once authorization is granted, modification to the affected tools is required to recognize new helper functions. This is an intentional design restriction. The tools must be carefully configuration managed so that one helper function does not interfere with another and so that a helper function does not interfere with the consistent operation of the tools across all segment types.

Not all of the tools listed in this appendix accept helper functions. Helper functions are only useful when processing segment descriptors during the development process, or during the installation process. Thus, tools such as `VerifySeg` and `COEInstaller` have helper functions, while tools such as `COEMsg` do not.

C-1.4 Summary Tables

Table C-1 and Table C-2 below are an abbreviated list of the tools that indicates the following:

- which tools have a library interface as well as exist as an executable
- which tools accept helper functions
- which tools are available only on NT platforms
- which tools are available only on Unix platforms.

If a tool is not listed, it means the following:

- the tool is available as an executable only; it does not have a library interface
- the tool does not accept a helper function
- the tools is available for both NT and Unix platforms.

Tool	Library I/F Avail	Helper Fnct Avail	Supported Platforms
1. COEAddDBUser	Y		U
2. COEAskUser	Y		*
3. COEBackupDB	Y		U
4. COEConnectAS			U
5. COECreateAS			U
6. COECreateDS	Y		U
7. COEDeleteDBUser	Y		U
8. COEDropDS	Y		U
9. COEExtendDS	Y		U
10. COEFindData	Y		N
11. COEFindDrive	Y		N
12. COEFindServer	Y		U
13. COEInstError	Y		*
14. COELstDBDepends			U
15. COELstDBRoleUsrs			U
16. COELstProfileSegs			U
17. COELstProfileUsrs			U
18. COELstSegProfiles			U
19. COELstSegUsrs			U
20. COELstUsrDBRoles			U
21. COELstUsrProfiles			U
22. COELstUsrSegs			U
23. COEMsg	Y		*
24. COEPrompt	Y		*
25. COEPromptPasswd	Y		*
26. COERebuildIndex	Y		U
27. COERebuildView	Y		U
28. COERemoveAS			U
29. COERestoreDB	Y		U
30. COEScanCOTS			N
31. COEStartDBServer	Y		U
32. COEStopDBServer	Y		U
33. COETestInstall		Y	*
34. COETestRemove		Y	*
35. COEUpdateHome	Y		U

Legend: **Y** - capability exists '**blank**' - no capability
U - Unix only feature **N** - NT only feature
* - All platforms

Table C-1: COE Runtime Tools

Tool	Library I/F Avail	Helper Fnct Avail	Supported Platforms
1. CanInstall		Y	*
2. ChkCompliance		Y	*
3. TestInstall		Y	*
4. TestRemove		Y	*
5. VerifyCOE		Y	*
6. VerifySeg		Y	*

Legend: **Y** - capability exists '**blank**' - no capability
U - Unix only feature **N** - NT only feature
* - All platforms

Table C-2: COE Developer Tools

That is, the tables list the tools only on an exception basis. As the legend indicates, a "Y" in the Library I/F column indicates that a library interface exists for the tool while a blank indicates it does not. Similarly, a "Y" in the Helper Fnct column indicates that the tool accepts a helper function while a blank indicates that the tool does not. An "N" in the Supported Platform column indicates that the tool is available for NT platforms only, a "U" indicates it is available for Unix platforms only, while an "*" indicates the capability is available for both platforms.

C-2. COE Runtime Tools

This section lists the COE tools that are available at run time. These executables are delivered to the operational site and are located underneath the directory `/h/COE/bin`.

C-2.1 COEAddDBUser

The creation and maintenance of COE database server user accounts are integrated with COE system management functions for creating and maintaining users' operating system accounts. Information on database users must be maintained so that they may be restored when a data store is restored. Restore files must be built by this application.

This function supports the discretionary granting of access within the database. It synchronizes users' permissions inside the database with their application profiles outside the database. It uses the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

This function is accessed automatically when accounts are added with profiles that include database access. It may also be accessed independently as part of the Database Administrator account group.

C-2.2 COEAskUser

This tool is intended for use in the `PostInstall` script to display a message to the user and have the user click on a "Yes" or "No" button. The syntax is

```
COEAskUser {parameters} msg
```

where *msg* is the prompt to display to the user. The parameters allow the calling routine to specify the labels to be used for the "Yes/No" buttons.

C-2.3 COEBackupDB

The standard dump and restore format for the RDBMS will be provided. Special format data backup options will also be available as specified by the data store segments. This feature is available from the Database Administrator account group.

C-2.4 COEChkUpdates

The `COEChkUpdates` tool is invoked from the System Administration account group. It allows specification of a "master" machine to be checked for the availability of new segments, segment updates, or segment patches. The checking may be done on demand, or at a specified interval (e.g., every 24 hours). The default operation is to check on demand.

When `COEChkUpdates` determines that system updates are available, a notification message is displayed to the logged in operator that new features are available. A list is *not*

displayed, only a notification is presented. The system administrator may then log in, invoke `COEChkUpdates` on demand, and receive a list of available new features for downloading.

C-2.5 COEConnectAS

`COEConnectAS` connects a client workstation to an application sever. This tool is selected from the System Administration account group. The client workstation must have at least the COE kernel installed, but may also have other segments loaded locally to improve performance. By default, `COEConnectAS` does not load segments from the application server onto the client workstation. They remain on the application sever and are loaded into the client workstation's memory as required for execution. However, `COEConnectAS` provides a feature that allows dynamic loading of segments. This feature allows segments to be loaded onto the client workstation disk the first time an operator attempts to accesses a function in the segment. Future accesses are then done from the client workstation's local copy. Dynamically loaded segments are removed from the client workstation when the operator logs out, and a check is made each time a new operator logs in to a "dynamic" workstation to be sure that segments were not inadvertently left on the client workstation.

C-2.6 COECopyWS

`COECopyWS` is provided as an aid to speed up the site installation process. It allows segments on one workstation, the source workstation, to be copied across the network onto another workstation, the target workstation. This command will cause the target workstation to be configured identically to the source workstation, including ensuring that the kernel COE is installed and configured. Workstation-specific items, such as the target workstation's hostname and IP address are unmodified. Disk partitions and other actions normally performed when the bootstrap COE is loaded are not affected on the target workstation.

The source workstation must already have all segments installed that are to be copied to the target. The target workstation must have at least the bootstrap COE installed, and must be reachable across the LAN from the source workstation. The target workstation must have sufficient disk space to hold all of the segments that are to be transferred. `COECopyWS` checks to be sure that this is the case, and reports an error message and aborts if it is not.

`COECopyWS` does not require that the target and source have identical disk partition configurations. However, segments that are grouped together on the same disk partition on the source workstation will be grouped together on the same disk partition on the target workstation.

This command is normally selected from the System Administration menu, but it may also be invoked from the command line. It may be executed only from the source workstation.

C-2.7 COECreateAS

`COECreateAS` allows a site administrator to create an application server. It is normally selected from the System Administration account group and operates the same way as the `COEInstaller` tool. The difference is that this tool only loads the segments onto disk to be executed by another platform (see `COEConnectAS`), and this tool allows multiple hardware types to be loaded on the application server.

The `COECreateAS` tool does *not* support installation of multiple versions on the application server. This could lead to problems if two different versions of a segment, for the same platform type, are executed at the same time.

C-2.8 COECreateDS

`COECreateDS` creates data stores on a COE Database Server and allocates physical storage to the data stores. It also creates the database owner account associated with the data stores. `COECreateDS` is also used to create a new data store for an existing database owner. It must be executed when the database server is running. The purpose of `COECreateDS` is to isolate the database segment developer from the physical storage implementation of the various DII database server configurations.

Should `COECreateDS` encounter an error it cleans up any files, accounts, or other objects that were created during processing.

C-2.9 COEDeleteDBUser

This function supports the discretionary revocation of access within the database. It synchronizes users' permissions inside the database with their application profiles outside the database. It uses the database role information under the application segment's `Database` descriptor in `SegInfo` to do this.

This function is accessed automatically when accounts are deleted with profiles that include database access. It may also be accessed independently as part of the Database Administrator account group.

C-2.10 COEDelUnused

This tool examines all of the installed segments and determines which are unused (e.g., not in any profile, no other segment is dependent upon it). The list is presented to the system administrator who may then delete one or more of the segments. The purpose of this tool is to aid site administrators in finding and removing segments that are not be used in order to free up disk space.

This command is normally selected from the System Administration menu, but it may also be invoked from the command line. It may be executed from one workstation to examine itself or another workstation.

C-2.11 COEDropDS

`COEDropDS` removes the files associated with an existing data store from the database server. It is intended to be executed after all data objects have been removed using a database segment's `DEINSTALL` script. Accordingly, `COEDropDS` will fail if any data object exists in the data store being removed. `COEDropDS` cannot be used to remove components of the DBMS Server.

If neither a store name nor a store list is provided, `COEDropDS` will attempt to remove all data stores associated with the specified DBO and then to remove the DBO account from the DBMS. Should `COEDropDS` encounter an error it will restore any files, accounts, or other objects that were deleted during processing.

C-2.12 COEExtendDS

`COEExtendDS` adds additional physical storage to an existing data store. For `COEExtendDS` to be used the database owner account and the data store name must both exist. If the intent is to add a new data store (e.g. additional "Sybase segment") to an existing DBO, then `COECreateDS` should be used.

Should `COEExtendDS` encounter an error during processing it cleans up any files or other objects that were created before the error occurred. Also, the API will return an error status to segment installation script.

C-2.13 COEFindData

`COEFindData` is an NT only tool. Its purpose is to determine where a data segment was loaded. The syntax is the same as for `COEFindSeg`. It is normally invoked from a `PostInstall` or other installation-related script.

C-2.14 COEFindDrive

`COEFindDrive` is an NT only tool. The syntax is the same as for `COEFindSeg`. It returns the name of the disk drive, which may be a network drive, that contains the specified segment.

C-2.15 COEFindSeg

This tool returns information about a requested segment. The tool accepts as input the name of the directory where the segment is expected to be, the segment name, and the segment prefix. If the directory is omitted, or the expected directory is not there, the tool performs a search for the segment in the legal directory locations for segments identified in Chapter 5 (e.g., `/h`, `/h/AcctGrps`, `/h/COE/Comp`).

The tool returns the absolute path of the directory where the segment was found, the segment name found, the segment prefix, and the segment type, including the segment attribute if applicable.

C-2.16 COEFindServer

`COEFindServer` determines the hostname, IP address, and whether or not the server is a DCE server, for the requested server. The server name is the name specified by the `$SERVERS` or `$DCESERVERS` keyword in the Network descriptor. If the server is a DCE server, the hostname and IP address returned will be of the first host where the server is found. There may be multiple hosts running the requested server.

C-2.17 COEInstaller

`COEInstaller` is normally executed from a System Administrator menu by an operator who has system administration privileges. It displays a list of configuration definitions or segments that may be installed from tape, disk (e.g., a network segment server), or other electronic media. It may also be executed directly from the command line.

By default, `COEInstaller` does not write any output to `stdout`. The verbose and warning flags are both disabled unless enabled by the `-v` and `-w` parameters respectively.

The `COEInstaller` writes information to a status log that indicates installation progress, which segments have been installed, and other information that might be useful to the site administrator. The site administrator may also choose to have the `COEInstaller` log information that is generated from the `PreInstall`, `PostInstall`, and `DEINSTALL` scripts. That is, the output from these scripts is "piped" to the status log. This capability is useful in the event problems are encountered during the installation process. The `COEInstaller` also provides an option to print the status log.

C-2.18 COEInstError

`COEInstError` allows a segment to display an error to the user from within a `PreInstall`, `PostInstall`, or `DEINSTALL` script and terminate installation of the segment. `COEInstError` *always* returns the same return code and will cause `COEInstaller`, `TestInstall`, and `TestRemove` to halt further processing of the segment.

If the calling descriptor (e.g., `PreInstall`, `PostInstall`, or `DEINSTALL`) does execute `COEInstError`, `COEInstaller` will detect that `COEInstError` has been called, terminates segment installation, and will display an appropriate error message. Whether or not the calling script sets a "terminate" return code, the `COEInstaller` will detect that `COEInstError` has been called and terminate segment installation.

`COEInstError` must be used carefully. If it is invoked from within `PreInstall` or `PostInstall`, the installation tools will remove the segment because the installation has been unsuccessful. If invoked from within `DEINSTALL`, the installation tools will *not* remove the segment.

C-2.19 COEListDepends

COEListDepends is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments upon which a specified segment depends. If no segment name is specified, a GUI is displayed and the user is prompted to select a segment from a list of segments accessible on the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.20 COEListSegs

`COEListSegs` is normally selected from a System Administration menu. It displays a sorted list of all segments that have been installed on a specified machine. The tool may be run in command-line mode only (with output written to `stdout`), or through a GUI.

C-2.21 COELstDBDepends

`COELstDBDepends` executes within the DBMS to search object dependencies across multiple database segments. Depending on the input parameters, it will search for all dependencies of a specified data object or for all dependencies of a database owner's schema on other schema's objects.

C-2.22 COELstDBRoleUsrs

`COELstDBRoleUsrs` provides a sorted list of all user's database login accounts assigned to a specific database role. This function is available from the Database Administrator account group.

C-2.23 COELstProfileSegs

`COELstProfileSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments accessible from a specified profile. If no profile is specified, a GUI is presented from which the user can select a profile. If a profile is specified, it is assumed that the tool is being run from a command line and all output is sent to `stdout`.

C-2.24 COELstProfileUsrs

`COELstProfileUsrs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts

(designated as local or global) that are assigned to a specific profile, where the profile may be global or local.

If no profile is specified, the user is prompted to select a profile from a list of profiles available on the machine. If a profile is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.25 COELstSegProfiles

`COELstSegProfiles` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group that contain a specified segment.

If no segment name is specified, the user is prompted to select a segment from a list of segments accessible from the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.26 COELstSegUsrs

`COELstSegUsrs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all user login accounts (designated as local or global) that have access to a given segment.

If no segment name is specified, the user is prompted to select a segment from a list of segments accessible from the machine. If a segment name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.27 COELstUsrDBRoles

`COELstUsrDBRoles` provides a sorted list of all database roles assigned to a specific user. This function is available from the Database Administrator account group.

C-2.28 COELstUsrProfiles

`COELstUsrProfiles` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a list of all profiles sorted by account group that are assigned to a specific user.

If no user login account name is specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. If a user login account name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.29 COELstUsrSegs

`COELstUsrSegs` is normally selected from a System Administration menu. It may also be executed from a command line. It displays a sorted list of all segments that are accessible by a specific user.

If no login account name is specified, the user is prompted to select a login account from a list of local and global accounts accessible from the machine. If a login account name is specified, it is assumed that the tool is being run from a command line and the tool writes the output to `stdout` only instead of a window.

C-2.30 COEMsg

This tool is intended to be used by `PreInstall`, `PostInstall`, and `DEINSTALL` to display an informational message to the user. A message is displayed in a window and the user must click on an "OK" button to continue.

C-2.31 COEPrompt

This tool is similar to `COEMsg`, but expects the user to enter a response. The calling routine may indicate the maximum number of characters in the user's typed response. The default, if not specified, is 40 characters.

C-2.32 COEPromptPasswd

`COEPromptPasswd` is similar to `COEPrompt` in syntax and operation. It is intended to be used in `PreInstall` and `PostInstall` to prompt a user to enter a password. This routine displays a window in which the user may enter a password. The user's response is not echoed in the window.

By default, `COEPromptPasswd` will prompt for the password, then prompt the user to enter the password again for confirmation. A flag may be passed to the tool to indicate that a confirmation prompt is not desired. The default maximum password length that a user can enter is 14 characters, while the minimum is 6 characters. As with `COEPrompt`, a parameter to this tool allows the minimum and maximum to be set to other values.

When confirmation is requested, the tool prompts the user for a matching confirmation up to 3 times before returning a failure. If after 3 attempts the confirmation does not match, the tool returns an error code and it is up to the calling routine to determine what action to take. This works the same way whether the tool is invoked via a library interface, or as a separate executable program.

C-2.33 COEPropagateUpdates

This tool is available only to the system administrator. It allows segments (new segments, upgrades, and patches) to be sent to other workstations. It is used in conjunction with

COEChkUpdates to automatically update workstations on the LAN as new upgrades are available.

One possible scenario is to use COEChkUpdates to determine that new segments are available. Then, the segments are downloaded to a network installation server and temporarily installed (COETestInstall) for testing. Once verified as acceptable at the site, COEPropagateUpdates can be used to determine which workstations on the LAN need to be upgraded and then automatically perform the upgrades.

C-2.34 COERebuildIndex

COERebuildIndex is used to recover or restore corrupted indexes within the database server. It recreates the specified index and is available from the Database Administrator account group.

C-2.35 COERebuildView

COERebuildView is used to recover or restore corrupted views within the database server. It saves any existing grants to users or database roles and then recreates the specified view. It then re-executes the saved grants to finish restoring the view. This function is available from the Database Administrator account group.

C-2.36 COERemoveAS

This tool removes a segment from an application server (see COECreateAS). COERemoveAS is selected from the System Administration account group. When all segments in a particular hardware type are removed, all directories associated with the hardware type on the application server are removed.

C-2.37 COERestoreDB

The restore will not only restore the backup data store (data and structure), it will also run the Restore scripts for other data store segments to complete the restore process. The owning segment will tag the Restore script with ".main". This script will be run first, followed by other public data stores, and then private data stores which have Restore scripts for the data store being restored.

This feature is available from the Database Administrator account group.

C-2.38 COERmtInstall

COERmtInstall is the remote installation tool. It is normally selected from the System Administration menu, but may also be invoked from the command line. The tool operates in either a "push" or a "pull" mode.

Push Mode

Push mode operation consists of an operator at a repository site (e.g., DISA Operational Support Facility) sending one or more segments from the repository site to the remote site. The "pushed" segment is sent to a remote network's installation server and when the transfer is completed, the tool can be asked to automatically invoke the `COEInstaller` tool on the remote machine to allow the operator from the sending site to perform an actual installation.

Pull Mode

Pull mode operation consists of an operator at a remote site requesting the transfer of one or more segment install file(s) from the repository site to the remote site. In pull mode, the "pulled" segment is loaded onto the remote site's installation server from which the operator may install the segment on one or more machines.

Remote Segment Removal

`COERmtInstall` also allows a repository site to remotely remove a segment on a machine at the remote site. This is essentially accomplished by launching `COEInstaller` on the remote machine and using it to select the segment(s) to delete.

`COERmtInstall` provides several security measures (encryption, passwords, anonymous ftp, etc.) to protect segment transmission and to prevent unauthorized access to the repository or a remote site. Discussion of such measures is beyond the scope of this document.

C-2.39 COEScanCOTS

The `COEScanCOTS` tool is available only on NT platforms. It scans the disk to find applications that are already installed and automatically creates segment descriptors for them. This allows the COE to be loaded on an already established NT environment where several COTS products may already have been loaded. Then, with segment descriptors created for these pre-loaded COTS products, segments loaded on later can express dependencies on them.

C-2.40 COEStartDBServer

`COEStartDBServer` will check to see if the specified server is running and if not, it will start the DBMS to support the installation of the a database segment.

> **Note:** This tool starts the database server in its maintenance mode.

C-2.41 COEStopDBServer

`COEStopDBServer` supports the shutdown of the DBMS as necessary to support the installation of a data base segment. The DBMS server will be located by the supplied name using the interfaces and hosts file and stopped using the DBMS commands for a normal shutdown of the server.

C-2.42 COETestInstall

The `COETestInstall` operates exactly like the normal `COEInstaller`, except that its purpose is to temporarily install a segment on a workstation. It is normally selected from the System Administration account group, but may be executed from a command line. The temporary installation may be an individual segment, or a collection of segments from a configuration definition. The purpose of a temporary install is to allow a site administrator to temporarily install a segment on an isolated test workstation for the purposes of checking out the segment before committing it to widespread installation.

When a segment is temporarily installed, an expiration date may be entered. The default is 48 hours. When the expiration date arrives, a warning message is displayed to the operator upon login to indicate that a segment has expired. However, the segment is *not* removed and the notification is for warning purposes only. `COETestRemove` must be invoked to actually remove the segment. If the test is successful, the `COETestInstall` tool can also be used to "promote" the segment from test status to "installed" without the need to delete and reinstall the segment.

C-2.43 COETestRemove

`COETestRemove` is selected from the System Administration account group and is used to remove a segment that was temporarily installed. Its interface looks the same as the `COEInstaller`, but it only lists temporarily installed segments and will not allow the removal of any other type of segment.

Removal of a temporarily installed segment can be done at any time, whether or not the expiration has occurred. The operator has the option of either removing the segment, or making the installation "permanent." When a temporary segment is removed, the segment it replaced, if any, is restored.

C-2.44 COEUpdateHome

`COEUpdateHome` is intended to be invoked from within a segment's `PostInstall` script. Its purpose is to update the home environment variable within a script file to point to where a segment was actually installed.

This tool searches the named script file for the first place that the environment variable is defined through either `set` or `setenv`, and replaces the definition with the absolute pathname for where the segment was loaded.

For example, assume `COEInstaller` loads `MySeg` underneath `/home4/MySeg`, and the script file `.cshrc.MYSEG` contains the following:

```
setenv MYS_HOME        /h/MySeg
```

Then the statement

```
COEUpdateHome Scripts/cshrc.MYSEG MYS_HOME
```

changes the statement to read

```
setenv MYS_HOME        /home4/MySeg
```

Note: Since environment extension files are not required in the NT COE, `COEUpdateHome` is not available for NT platforms. `COEFindData` and `COEFindDrive` are available to perform a similar function for NT platforms.

C-3. COE Developer Tools

This section lists the COE tools that are available during development, but are not delivered to operational sites. By default, these executables are located underneath the directory `/h/TOOLS/bin` and are distributed as part of the Developer's Toolkit. These tools are not location sensitive, and may be moved to any directory desired for development.

C-3.1 CalcSpace

`CalcSpace` computes the space required for the segment specified and updates the `Hardware` segment descriptor accordingly. The segment referenced must *not* be compressed, and must not contain any files that do not belong with the segment (e.g., source code) at runtime. Otherwise, the space computed will be incorrect.

The `-v` flag enables the verbose flag and causes the tool to print out the space requirements for each top-level subdirectory. The warning flag is enabled by default and causes the tool to print a warning message if directories are found that are not expected (e.g., `include`, `src`) or if expected directories are missing for the segment type (e.g., `bin` for software segments). `CalcSpace` does not affect the reserve space request in the `Hardware` descriptor, and does not affect any partitions specified.

The amount of space required is reported in K bytes.

C-3.2 CanInstall

`CanInstall` tests a segment to see if it can be installed. It performs the same tests that `COEInstaller` does at installation time. To be installable, all required segments must already be on the disk, and there must not be any conflicting segments on the disk.

C-3.3 ChkCompliance

The `ChkCompliance` tool examines a segment to determine its Category 1 compliance level. Some requirements cannot be checked automatically (e.g., using only POSIX calls, expected location of X/Motif libraries), so the tool reports the *maximum* possible level of compliance based on the criteria that can be checked automatically.

C-3.4 ChkDBCompliance

`ChkDBCompliance` examines a database segment's database to determine its level of compliance. It executes within the DBMS to complement `ChkCompliance`. It cannot be executed by independently, because it is a helper function for `ChkCompliance` and is called automatically from `ChkCompliance`.

C-3.5 CMMakeDistrib

This tool is intended for use by Configuration Management departments. It is a configurable tool that allows groups to be defined (e.g., operational sites, developers) and to package all or portions of a submitted segment for delivery to the selected group.

For example, an operational site does not need libraries and header files but developers do. Developers require these to be able to create application segments, but source code is not normally sent to developers for packages written by another developer. For the operational site, `CMMakeDistrib` would ensure that header files and libraries are excluded from the delivery, but are present for the package sent to a developer. `CMMakeDistrib` extracts the necessary directories out of the repository and provides them to `MakeInstall` for creating the actual distribution media.

C-3.6 COEIDE

The `COEIDE` (COE Integrated Development Environment) tool provides a GUI for accessing all of the developer tools in subsection C-3, and any runtime tools (such as `COEInstaller`) that are useful during the development process. The tools accessible from `COEIDE` are limited to those which operate on segments, such as `VerifySeg` or `MakeInstall`, but not runtime tools such as `COEMsg` that are invoked by the segment itself.

`COEIDE` is a menu-and-icon-driven interface that allows the developer to open segments for development work. Once opened, any of the development tools can be run against the segment. `COEIDE` allows the output from tools to be captured into a log file for later printing or review. It also includes the ability to set tool flags such as the verbose and warning flags.

The purpose of `COEIDE` is to provide a simplified, single interface to all of the tools that are useful to create segments or retrieve them from a repository, validate them, test them, and then submit them electronically to the appropriate SSA. In this case, the SSA could be within the developer's own organization so that the tool can be used to transition a segment from the developer to configuration management, and to integration and test organizations.

`COEIDE` is available for both the Unix and the NT environments.

C-3.7 ConfigDef

`ConfigDef` is used to create a configuration definition from a list of segments. Components of a configuration definition (e.g., bundles, configurations, folders) may participate in more than one configuration definition.

Segments will normally reside in a segment repository such as SDMS. Thus, `ConfigDef` provides an interface that allows segments to be checked out of the repository. The interface is the same as that provided by `MakeInstall`.

C-3.8 ConvertSeg

`ConvertSeg` is a tool that examines segment descriptors and converts them to the latest format. To the extent possible, obsolete usage is identified and corrected as part of this process. The original segment descriptor directory is not modified, but it is renamed to be `SegDescrip.orig`. The converted segment descriptors created by the tool are placed in a newly created (e.g., after the original `SegDescrip` is moved to `SegDescrip.orig`) `SegDescrip` directory. To avoid inadvertently overwriting a segment descriptor directory, a prompt is issued if the directory `SegDescrip.orig` already exists.

This command is useful in combining individual segment descriptor files into the proper `SegInfo` descriptor. The tool will print a warning if obsolete directories, such as `progs` and `libs`, are encountered.

C-3.9 MakeAttribs

This tool creates the descriptor file `FileAttribs` described in Chapter 5. It recursively traverses every subdirectory beneath the segment home directory and creates a file with lines in the format

```
permits:owner:group:filename
```

At installation time, the installation tools perform the following statements for each entry:

```
chmod permits $INSTALL_DIR/filename
chown owner $INSTALL_DIR/filename
chgrp group $INSTALL_DIR/filename
```

For security reasons, `MakeAttribs` does *not* include any file owned by root nor any file for which *permits* is greater than 777. A warning is printed for each filename that is rejected. A warning is also printed for each "suspicious" permission setting such as 777 for a file, execute permissions on files in a `data` subdirectory, etc.

C-3.10 MakeInstall

The `MakeInstall` tool writes one or more segments to an installation medium, or packages the segments for distribution over SIPRNET. `MakeInstall` checks to see if `VerifySeg` has been run successfully on each of the segments, and aborts with an error if it has not. `MakeInstall` supports multi-volume tapes, but will not split segments across tapes. It allows segments for different hardware platforms to be on the same

installation medium, and it also allows configuration definitions to be written to the installation medium.

Command-line parameters and segment lists may be repeated as required. A command-line parameter applies to all succeeding command-line arguments until the next parameter is encountered.

All components of an aggregate must be included at the same time. The order in which segments are passed to `MakeInstall` is unimportant because the tool will order them in such as way as to guarantee that the tape will not need to be rewound during installation.

`MakeInstall` requires that segments be available on disk with at least the `SegDescrip` subdirectories in an uncompressed, unencrypted format. Since segments are normally installed in a repository such as SDMS, `MakeInstall` provides an interface to allow segments to be checked out from the repository as required. This is done by invoking a user supplied program and passing it the name of the segment to extract. The supplied program checks the requested segment out of the repository and places it in an agreed-upon temporary location. `MakeInstall` automatically deletes the temporary segments when finished with them.

C-3.11 mkSubmitTar

Segments must be packaged by compressing and encrypting them prior to submitting them electronically to SDMS. `mkSubmitTar` does this task. It checks to see if `VerifySeg` has been run successfully on the segment and aborts with an error if it has not. `mkSubmitTar` also ensures that the directories defined in Chapter 5 as required for segment submission are provided.

`mkSubmitTar` allows segments to be checked out of a repository prior to packaging for submission. `mkSubmitTar` will only package one segment into a tar file. Each component of an aggregate is packaged separately. However, `submit` allows multiple files to be sent during one session. An entire aggregate should be sent at one time unless the size is prohibitive.

C-3.12 MoveSeg

`MoveSeg` allows a segment that has already been installed to be moved from one location to another, including to another disk partition. An error will be generated if there is not enough room to move the segment. The old segment location is deleted when the move is completed.

C-3.13 SegAnalyze

`SegAnalyze` analyzes the segment specified and creates a table showing total memory and disk usage. It determines all segment dependencies so that the statistics reported include all required segments.

Note: SegAnalyze was called VarAnalyze in the previous *I&RTS*.

C-3.14 SegCatalog

This tool adds a segment to the segment catalog.

C-3.15 submit

This tool electronically transmits segments packaged by mkSubmitTar to the host repository. By default, segments are *not* deleted from the submitting machine after transmission.

C-3.16 TestInstall

TestInstall is used to temporarily install a segment that already resides on disk. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. VerifySeg must have been run before TestInstall to make sure that the segment is valid.

TestInstall performs the same operations as COEInstaller except that it does not need to read the segment from tape (e.g., it is already on disk), and the segment may be in any arbitrary location. TestInstall will establish the required symbolic link under /h to preserve the COE standard directory structure.

C-3.17 TestRemove

TestRemove is used to remove a segment that was installed by TestInstall. It must be run when there are no other COE processes running. The reason for this restriction is that the tool may modify COE files already in use with unpredictable results. TestRemove removes the symbolic link under /h if one exists, but it does *not* delete the segment from disk.

Note: TestRemove is unconditional and should be used with great caution. It will remove a specified segment even if other segments still installed depend upon it.

C-3.18 TimeStamp

TimeStamp puts the current time and date into the VERSION segment descriptor. It is intended to assist the configuration management process by allowing the time stamp to be updated just prior to running VerifySeg and mkSubmitTar for the deliverable product.

C-3.19 unpackSubmitTar

This tool will decompress and decrypt segments that have been packaged with the `mkSubmitTar` tool.

C-3.20 VerifyCOE

The tool `VerifyCOE` is used to validate the COE runtime environment. It is intended to be used after the kernel COE has been loaded and configured to ensure that it has been correctly set up. This tool is primarily intended for developers who wish to alter the COE installation instructions to conform to development environment needs.

Specific tests performed include:

- Check for sufficient swap space on the disk.
- Check for correct disk partitioning.
- Check for expected device drivers.
- Check for sufficient shared memory, message pool size, and other operating system kernel parameters.
- Check for proper location of X and Motif libraries.

C-3.21 VerifySecurity

The `VerifySecurity` tool is used to perform security checks against a segment (similar to the operation of `VerifySeg`), against an installed COE (similar to `VerifyCOE`), or against an installed system. A single security policy cannot be stated for all COE-based systems, so this tool is limited to identifying potential security risks that are common across all systems (e.g., files that are world readable/writeable, use of remote commands, improper .rhosts files, etc.). When used against an entire system, the tool interfaces to commercially available tools such as `Satan` to identify vulnerability areas. Because of the serious potential for virus attacks, especially in the NT world, the tool also performs virus checking through an interface to commercially available virus checkers.

`VerifySecurity` checks security-related issues to the extent possible. Neither it nor any other tool is fully comprehensive, nor does it obviate the need for a disciplined security policy and security enforcement procedures.

C-3.22 VerifySeg

`VerifySeg` is used to validate that a segment conforms to the rules for defining a segment. It uses information in the `SegDescrip` subdirectory and must be run whenever the segment is modified. `VerifySeg` must be run for each segment. If the segment is an aggregate segment, `VerifySeg` must be run on each segment in the aggregate.

Specific tests include:

- Check to ensure that executables are prefaced with the segment prefix where required.
- Check to ensure that all defined environment variables use the segment prefix.
- Check to ensure that all pathnames are defined relative to the home environment variable, `segprefix_HOME.`
- Check to ensure that all required environment variables (`USER_HOME`, `USER_DATA`, `DATA_DIR`, etc.) are defined by account group segments.
- Check to ensure that all required environment variables are defined by the COE parent component segment.
- Check aggregate segments for internal consistency (e.g., all components are hardware compatible).
- Check and warn if `mv` is used in a `PostInstall`, `PreInstall`, or `DEINSTALL` script since this may be an illegal attempt to move files across disk partitions.

C-3.23 VerifySegDB

`VerifySegDB` is used to validate that a database segment's database conforms to the rules for defining a segment's database. It executes within the DBMS and is a helper function for `VerifySeg`. It cannot be executed independently of `VerifySeg`, but is called automatically from `VerifySeg`.

C-3.24 VerUpdate

`VerUpdate` updates the segment version number, date, and time in the `VERSION` descriptor. Command-line parameters can be used to indicate which digits (e.g., major release, minor release, maintenance digit, or developer digit) to update. Multiple digits may be updated at one time. By default, only the maintenance digit is updated.

If no version number is specified, the version number inside the `VERSION` descriptor is incremented. That is, 1.0.0.0 is updated to be 1.0.0.1. If no version is specified, and `VERSION` does not exist or has no version number, the file is created and version number 1.0.0.0 is inserted.

> **Note:** No error checking is performed on the version number specified since the segment must still be validated by `VerifySeg`.

This page is intentionally blank.